Handbook of Constraint Programming

Edited by F. Rossi, P. van Beek and T. Walsh

Elsevier

Foreword

Constraints are an ubiquitous concept, which in its broader sense pertains to every day experience: they represent the conditions which restrict our freedom of decision. In fact, how much our choices are constrained by the external world is a basic philosophical question. In the formalized reasoning of scientific disciplines, constraints have been employed extensively, from logic to numerical analysis, from mathematical programming to operations research. In computer science, constraints have been with us from the early days, for modeling, representing and reasoning (see the interesting historical remarks in Chapter 2 of this handbook, Constraint Satisfaction: An Emerging Paradigm).

I see several good reasons for this ubiquity: one is the conceptually clear separation between the perfectly declarative problem statements and the often cumbersome enumerative efforts for finding solutions. Another reason is the complexity challenge: the classical constraint satisfaction problem is NP-complete and in fact tautology checking in propositional calculus (a constraint problem on Boolean variables) has been the touchstone for this complexity class. A further reason is that large, complex constraint problems often occur in practice, they must be solved in one way or another, and fast, efficient, systematic solutions have an enormous economic value.

What I find surprising about constraints is that within artificial intelligence and computer science a relatively recent, relatively uniform body of knowledge has emerged which often yields decisive advantages over classical, extensively studied and well developed techniques. As for many success stories within computer science, success is largely due to a mixture of structures, algorithms, languages, programming techniques and system implementations. The aim of this handbook is to present this knowledge in all its facets. Different chapters are largely self contained and all contribute to put the subject into focus, similarly to the Hawaii Keck observatory, where the mirror is composed of 36 hexagonal segments.

From the conceptual point of view, the main characteristic features of constraint programming are constraint propagation, and the identification of various special cases which make complexity tractable. The former (see Chapter 3) is an inference technique which makes local constraints stronger without changing the global constraint. The latter issue concerns both the structure (see Chapter 7, Tractable Structures for Constraint Satisfaction Problems) and the kind of constraints (see Chapter 8, The Complexity of Constraint Languages). Less specific, but still very important issues are as follows: Backtracking Search Algorithms, in Chapter 4; Local Search, in Chapter 5; Global Constraints, in Chapter 6; Symmetry in Constraint Programming, in Chapter 10; and Modelling, in Chapter 11.

Another surprising fact about constraint theory is the incredibly close relationship with logic programming. In a rather precise sense logic programming is a way of expressing, and solving, certain classes of disjunctive, recursive constraints. Furthermore, logic programming can be very elegantly generalized to constraint logic programming (see Chapter

12), where the ordinary Herbrand constraint system, and its unification algorithm, are complemented with specific constraint solvers. The interaction with the committed choice languages studied in the Japanese projects of the eighties also yielded very interesting models of computation based on constraints. Amalgamation with more common (and efficiently implemented!) programming languages is also possible (see Chapter 13, Constraints in Procedural and Concurrent Languages).

Besides and beyond the beauty of its theoretical foundations, what contributes the most to the practical convenience of constraint programming are: (i) the development of specific results for important classes of constraints; (ii) the ability of extending the basic theory to various additional aspects which are very relevant in practice; and (iii) the flexibility and potential for integration with other modeling and solving methodologies.

About the development of specific results, this handbook includes chapters about constraints on finite (Chapter 14), structured (Chapter 17), temporal (Chapter 19), continuous and interval-based (Chapter 16) domains. The potential to extend the basic theory in evident in the case of soft constraints, considered in Chapter 9. Ordinary constraints are either satisfied or not, namely either true or false. Instead soft constraints return a more informative weight. Interestingly enough, the proposed extensions both accommodate several important cases (fuzzy, hierarchical, optimization, probabilistic constraints), and still often exhibit essentially the same solution algorithms. Extensions to random, changing and distributed/open constraints are treated in Chapters 18, 21 and 20 respectively.

About the last issue, in addition to the seamless integration with logic and imperative programming languages we mentioned already, quite remarkable are the paradigms resulting from the integration of constraint programming with operations research (see Chapter 15), with scheduling and planning (see Chapter 22), with vehicle routing (see Chapter 23), with component configuration (see Chapter 24), with (electricity, water, oil, data) networks (see Chapter 25), and with bioinformatics (see Chapter 26).

The global scenario based on service-oriented computing which is now under development offers additional theoretical and practical challenges to constraint programming. Conditions for service deployment and discovery, both functional and involving different aspects of quality of service, could be expressed in terms of hard and soft constraints, and the negotiation phases should involve substantial constraint solving abilities. Transactions among the various actors could also require partially backtrackable behavior or at least programmable compensations. Some level of real time, distributed, global constraint solving should be implemented in the middleware, since lots of higher level applications will be able to take advantage of, and pay for it.

I think that research and practical development in the area of constraint programming will be very active for quite a while in the future, establishing closer and closer connections with a variety of other design methodologies and even other disciplines. I consider this handbook not only a very nice piece of scientific work, but also a contribution quite instrumental at disseminating advanced knowledge about constraint programming both within the inner constraint community and across the much wider audience of potential users.

UGO MONTANARI
Dipartimento di Informatica
Università di Pisa

Contributors

Rolf Backofen
Albert-Ludwigs-Universität
Germany

Philippe Baptiste
CNRS LIX & École Polytechnique
France

Frédéric Benhamou
Université de Nantes
France

Christian Bessiere
LIRMM-CNRS
France

Kenneth N. Brown
Cork Constraint Computation Centre &
University College Cork, Ireland

Mats Carlsson
SICS AB
Sweden

David Cohen
Royal Holloway, University of London
United Kingdom

Rina Dechter
University of California, Irvine
USA

Boi Faltings
Swiss Federal Institute of Technology
Switzerland

Eugene C. Freuder
Cork Constraint Computation Centre &
University College Cork, Ireland

Thom Frühwirth
Universität Ulm
Germany

Ian P. Gent
University of St. Andrews
Scotland, United Kingdom

Carmen Gervet
Brown University
USA

David Gilbert
University of Glasgow
Scotland, United Kingdom

Carla Gomes
Cornell University
USA

Laurent Granvilliers
Université de Nantes
France

John N. Hooker
Carnegie Mellon University
USA

Holger H. Hoos
University of British Columbia
Canada

Peter Jeavons
University of Oxford
United Kingdom

Ulrich Junker
ILOG SA
France

Irit Katriel
University of Aarhus
Denmark

Philip Kilby
The Australian National University
Australia

Manolis Koubarakis
University of Athens
Greece

Philippe Laborie
ILOG SA
France

Claude Le Pape
ILOG SA
France

Alan K. Mackworth
University of British Columbia
Canada

Kim Marriott
Monash University
Australia

Pedro Meseguer
IIIA-CSIC
Spain

Laurent Michel
University of Connecticut
USA

Ian Miguel
The University of St. Andrews
Scotland, United Kingdom

Wim Nuijten
ILOG SA
France

Karen E. Petrie
University of St. Andrews
Scotland, United Kingdom

Jean-François Puget
ILOG SA
France

Francesca Rossi
Università di Padova
Italy

Thomas Schiex
INRA Toulouse
France

Christian Schulte
KTH - Royal Institute of Technology
Sweden

Paul Shaw
ILOG SA
France

Helmut Simonis
CrossCore Optimization
United Kingdom

Barbara M. Smith
Cork Constraint Computation Centre &
University College Cork, Ireland

Peter J. Stuckey
University of Melbourne
Australia

Edward Tsang
University of Essex
United Kingdom

Peter van Beek
University of Waterloo
Canada

Willem-Jan van Hoeve
Cornell University
USA

Mark Wallace
Monash University
Australia

Toby Walsh
National ICT Australia &
University of New South Wales
Australia

Contents

Part I

Foundations